Dokumentation

**Inhalt**

[**Vorwort** 2](#_Toc138400214)

[**Spielprinzip** 2](#_Toc138400215)

[**Vorbereitung** 2](#_Toc138400216)

[**Erster Plan** 3](#_Toc138400217)

[**Bibliotheken** 3](#_Toc138400218)

[**Aufteilung** 3](#_Toc138400219)

[**Kurzfassung des Verlaufs** 3](#_Toc138400220)

[**Programm** 3](#_Toc138400221)

[**Pakete** 3](#_Toc138400222)

[**Klassengruppen** 3](#_Toc138400223)

[**Klasse** 3](#_Toc138400224)

[**Ressourcen** 3](#_Toc138400225)

[**Sprites** 3](#_Toc138400226)

[**Sound** 3](#_Toc138400227)

[**Fazit** 3](#_Toc138400228)

[**Quellenangaben** 3](#_Toc138400229)

# **Vorwort**

Ein Spiel programmieren haben wir gedacht, es wäre einfach haben wir gedacht. Kurzgefasst, wir haben uns etwas Zuviel vorgenommen. Eine Mischung aus zu hohen Erwartungen, großen Ambitionen und etwas zuviel Kreativität bei der Planung haben uns gelehrt was es heißt, etwas selbst zu programmieren.

Dennoch wagen wir es zu behaupten, wir haben viel gelernt. Nicht nur, zu verstehen, dass der ein oder andere Selbstanspruch ungesund sein kann, sondern auch viel in der Programmierung selbst. So viel, wie in den letzten 1 ½ Monaten hatten wir in Java noch nie gelernt. Das ist unsere Odyssey vom ersten Planen   
bis zum (un)vollendeten Spiel.

# **Spielprinzip**

s

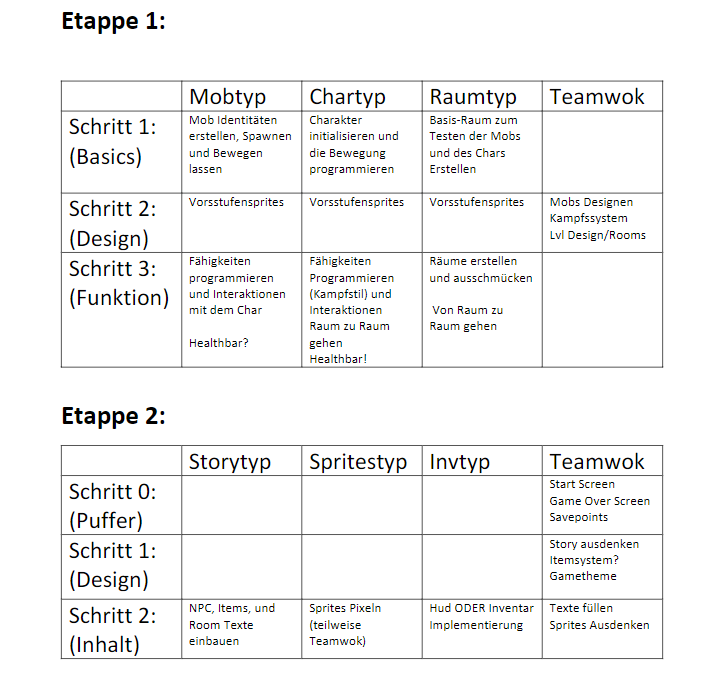
# **Vorbereitung**

Unsere Vorbereitungsphase begann direkt mit einer Überraschung. Durch das spontane Ausfallen eines Mitschülers betritt Jason Hein als drittes Mitglied des Projektes am Tag der eigentlichen Vorplanung. Daraufhin stellten wir folgenden Plan zusammen:

## **Erster Plan**

Nachdem wir uns eine erste grobe Idee erarbeiteten für unser Rogue-Like-Spiel teilten wir den Plan in ein Etappenprinzip ein.

**Jason**s Aufgabe (Mobtyp) war es sich um die Gegner zu kümmern, diese zu designen, implementieren und beweglich zu machen.   
**Marc**s Aufgabe (Chartyp) wiederrum war sich an den Spieler selbst zu wenden, diesen beweglich machen zu können und Fähigkeiten auszuführen  
**Leon** baute währenddessen das Spiel-interface und den Raum des Spiels selbst

**Erklärung:**  
Das eigentliche Ziel war es Etappe 1 abzuschließen. Diese wurde in 3 Schritten aufgeteilt. Schritt 1 stellte die Basics dar. Jeder sollte die ersten Objekte seiner Aufgabe erstellen sodass wenn Schritt 2 erreicht wurde, man durch simpelsten Sprites die Bewegungen auf dem Spielfeld sehen konnte. Schritt 3 würde dem Spiel dann letztendlich seine Funktion geben.

Etappe 2 & die derzeit undefinierte Etappe 3 wären zusätzliche Inhalte, doch wie so immer kommt es anders...

# 

# **Kurzfassung des Verlaufs**

Projektzeitraum: 15.5. - 30.6.2023

Nach der Aufstellung unseres ersten Plans trafen wir uns am 15.5. persönlich. Doch die ersten paar Stunden standen nicht unter einem guten Stern. Wir versuchten letztendlich vergebens JavaFX funktionabel für uns alle zum Laufen zu bekommen doch nach einigen Gesprächen stellten wir fest dass wir eine andere Lösung brauchten. Bessere Lösung ist bei Java.Swing leider nicht der Begriff der Wahl, aber der Vorteil ist eindeutig. Es funktioniert, und das konstant. Somit begann die Reise in den Codewahnsinn und wir fingen mit den ersten Zeilen an

Am Anfang arbeiteten wir eine kleine Klassenübersicht aus. Gemäß der Einteilung gab es die Klassen für die GUIs, SpielPanel, StartBildschirm und SpielFrame die hauptsächlich von Leon angegangen wurde. Recht schnell stand schon der kleine Startscreen, den man aus dem Programm kennt. Jason erstellte die Mob Klasse und implementierte ein KeyListener für die Steuerung. Marc kümmerte sich um die Charakter Klasse und dem Spieler seiner Cooldowns.

Grundsätzlich konnte man sagen das man voran kam. Die Übersicht Stand, problematisch war jedoch das zu oft separat voneinander programmiert wurde und Variablen und Funktionen, die eigentlich genormt werden konnten, nicht einheitlich bestimmt wurden. Später wurde das öfters zum Problem.

## **Bibliotheken**

Nach dem gescheitertem Versuch JavaFX zu benutzten, entschieden wir uns dagegen Aufgrund von 2 Gründen mit Java.Swing und teilen von java.awt zu Arbeiten. A) ist es standardmäßig im Programm vorhanden und braucht lediglich einen Import, somit ist es einfach an den Schulcomputern zu implementieren und B) gibt es mehr Möglichkeiten leichter und schneller die Bibliothek zu lernen durch ein besseres Angebot an Tutorials und Erklärungen.

**Importierte Bibliotheken:**

* Javax.swing.\*
* Java.awt.\*
* Java.awt.event.\*
* Java.util.\*
* Java.io.\*
* Java.net.URL
* Javax.sound.sampled.\*
* Javax.imageio.ImageIO

# **Programm**

## **Pakete**

***Main:***

***Block***:

### **Klassengruppen**

### **Klasse**

***Main(gemeinsam):*** Main ist das Haupt Programm, wo das Spiel startet. Es wird eine Instanz der Klasse StartBildschirm erstellt. Dannach wird eine Test Nachricht in der Konsole ausgegeben, um zu überprüfen, ob es keine Probleme gibt. Zusätzlich wird noch eine Instanz des SpielPanel erstellt und die Musik wird gestartet.

***StartBildschirm(Leon):***

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Die Swing Utilities Methode wird verwendet, um sicherzustellen, dass bestimmte GUI-Updates oder Operationen im Event Dispatch Thread (EDT) durchgeführt werden, der der Hauptthread ist, der für die Verarbeitung von Benutzeroberflächenereignissen verantwortlich ist. Der Hauptthread wird in SpielPanel verwendet. Es wird ein String erstellt mit den verschiedenen Sprachen, die man dann mit der Combobox in den Settings anpassen kann. Mit startBtn kann das Spiel gestartet werden. Die Abfrage actionPerformed überprüft ständig ob der Button gedrückt wird. Wird der Button gedrückt,wird durch dispose der frame des Startbildschirm geschlossen und das Spiel wird gestartet.

**SpielFrame(Leon):** Der Spielframe wird erstellt, sobald der Start Button in StartBilschirm gedrückt wird. Dieser Erstellt sofort das Notwendige Panel/SpielPanel.

Mit set title kann der name an dem oberen Rand eingetragen werden. Set resizable verhindert das vergrößern bzw das verkleinern des frames. Damit das Spiel sich an die Größe des Bildschirms anpasst, wird der Befehl setExtendedState verwendet und auf Maximum\_BOTH eingesetzt. Both steht dafür dass sowohl Horizontal als auch vertikal die gesamte Größe des Bildschirms verwendet wird. Zusätzlich startet das SpielFrame noch den gameThread mit der Methode startgameTread().

***SpielPanel(Gemeinsam):*** Damit der gameThread überhaupt benutzt werden kann, muss er noch Implementiert werden, dies wird mit dem Befehl **implements Runnable** gemacht. Der Thread ruft automatisch die Methode run auf, die eine Schleife enthält, die den Spielzustand aktualisiert und das Panel durchgehend neu zeichnet. Die Methode "update" wird verwendet, um den Spielzustand basierend auf Benutzereingaben (Tastatur- und Mausereignisse) zu aktualisieren. Innerhalb der Update Methode werden die Handler Klassen Mouse- und Keyhandler aufgerufen

***Mobs:***

***SpriteSheet:***

***Sound:***

***Handler:***

## **Ressourcen**

### **Sprites**

### **Sound**

# **Fazit**

# **Quellenangaben**

<https://youtu.be/om59cwR7psI>

<https://youtu.be/VpH33Uw-_0E>

<https://youtu.be/oPzPpUcDiYY>

<https://youtu.be/6Tj6XYGWfko>

<https://youtu.be/Kmgo00avvEw?t=8296>

<https://youtu.be/nUHh_J2Acy8>